



Stardock

DesktopX

Developer's Guide

Version 2.4

December 14th 2004

Copyright © 2004 Stardock.net, Inc.

Contents

1 Introduction	4
1.1 ABOUT THIS GUIDE.....	4
2 Development Basics	5
2.1 CREATING A THEME	5
<i>Hiding elements</i>	5
<i>Workarea</i>	5
<i>Resolutions</i>	6
<i>Wallpaper and Labels</i>	6
<i>Theme Information</i>	7
2.2 CREATING AN OBJECT.....	8
2.3 OBJECT TYPES.....	9
<i>Layer</i>	9
<i>Shortcut</i>	10
<i>URL</i>	10
<i>Object Controller</i>	11
<i>Taskbar</i>	11
<i>System Tray</i>	12
<i>System</i>	12
<i>DesktopX</i>	12
<i>Widget</i>	12
2.4 OTHER OBJECT BASICS	13
<i>Labels</i>	13
<i>Summary tab</i>	13
<i>Alignment</i>	14
2.5 EXPORTING WIDGETS, OBJECTS AND THEMES	15
3 Enhancing Objects.....	17
3.1 OBJECT APPEARANCE & STATES	17
<i>States</i>	17
<i>Image objects</i>	17
<i>Image sizing and offsets</i>	19
<i>Text Objects</i>	19
<i>Object Transparency</i>	19

	<i>Shadows and Glows</i>	20
	<i>Color</i>	20
	<i>Sound</i>	20
	<i>New States</i>	21
3.2	ANIMATING OBJECTS	22
	<i>Static Animations</i>	22
	<i>Image animations</i>	23
3.3	THE BASICS OF RELATIONSHIPS	24
	<i>Visibility</i>	24
	<i>Z-Order</i>	24
	<i>Movement</i>	24
	<i>Activation</i>	25
	<i>Start with</i>	25
	<i>Default Cursor</i>	25
	<i>Popups</i>	25
	<i>Position Adjustment</i>	26
3.4	GROUPS, PARENTS AND CHILDREN	27
	<i>Groups</i>	27
	<i>Parents and Children</i>	28
3.5	SENDING MESSAGES	29
3.6	CUSTOM FILES	30
3.7	DESKTOPX PLUGINS	31
	<i>Digital Clock</i>	31
	<i>Digital Clock (Text State)</i>	31
	<i>Email Notify</i>	32
	<i>Performance Meters</i>	32
	<i>DX Player</i>	32
4	Tips and guidelines	33
4.1	DO NOT ASSUME DEFAULT "START WITH" COMMAND	33
4.2	PLUGINS	34
4.3	CHILD OBJECTS	35
4.4	ANIUTIL WITH OPTIMIZATION FOR MAKING ANIMATIONS	36
4.5	32 BIT IMAGE FORMATS	36
4.6	SCREEN RESOLUTION ADAPTATION BEFORE DISTRIBUTING YOUR OBJECTS	36
4.7	JPG WALLPAPERS	36
4.8	METER OBJECTS	37
4.9	WRITING "PROPER" SCRIPTS	37
4.10	BUILDING "PROPER" OBJECTS BROWSING THE INTERNET	37

4.11	NEVER USE THE SLEEP METHOD IN SCRIPTS	38
4.12	ONLY PACKAGE USEFUL OBJECTS	38

1

Introduction

1.1 About this guide

Having mastered the use of DesktopX, you are probably ready to start creating objects for yourself. This may be just a simple object or you may already have grand plans based upon what you have seen other objects do.

This guide is designed to ease you through the process of creating objects. Don't feel that you have to take it all in at once. If you don't have the time to sit down and read this document then you should consider skimming the document to get an understanding of what it contains and then skip to the sections as you need them.

Scripting is a further advancement for creating even more sophisticated objects. The DX Scripting Guide describes this.

You should also be aware that in addition to this document, you can also obtain help in several other places:

The Stardock newsgroups are frequented by both Stardock staff and developers. They are therefore a great place to get advice and help. The news server is news.stardock.com and the primary group for DesktopX is stardock.desktopx

You can also find help via chat on irc.stardock.com in the #stardock channel.

This document will evolve with future releases. This document is written and maintained by Martin Conroy so if you have comments or suggestions, feel free to email me (martin@stardock.com), or visit us in the newsgroup or us on IRC chat.

Have fun!

2

Development Basics

2.1 *Creating a theme*

If you are planning to start working on DesktopX objects it is often easier to start with a blank canvas so you aren't distracted by other objects on your desktop.

The easiest way to do this is to select 'New Theme' from the 'Theme' tab on the DesktopX configuration dialog that you were introduced to in the User's Guide.

After choosing whether to save your existing theme, you will then be asked to confirm that you reset all of the existing theme's elements like wallpaper and work areas.

On the 'Theme' tab, you will find all the theme related items that you can define. These are ways of ensuring that the user experience of this theme will be consistent.

Hiding elements

You have two options here. 'Hide desktop icons' means that no icons will be displayed on the desktop, not even IconX objects. By doing this you can be sure that the desktop will be totally clear. You can also remove the taskbar from the desktop by checking 'Hide Windows taskbar'. This will mean that the user has no access to the start menu, running applications, or system tray unless you create objects to replicate these functions so you should use this option cautiously.

Workarea

This button allows you to fool Windows into thinking the desktop is actually smaller than it is. Why would you want to do this you may wonder? Well, one example is if you create objects along the edge of the screen, like the windows taskbar. If you do this without reducing the area then there are two things that occur; either maximized applications cover your objects so you can't see them, or your object covers the window so you can't see that. Obviously, you don't really want this so in this scenario. If you click the 'Work area' button and then check the 'Reduce work area' box, then enter a number in the 'Bottom' box to specify the number of pixels you want

to reduce the desktop by. Once you have done this then applications (when maximized) will not intrude into this space so your objects are visible.

Resolutions

There is also a 'Resolutions' button so you can specify the screen resolutions your theme is optimally designed to work with. DesktopX provides some advanced features to allow you to define how individual objects position themselves dependant on changes in the screen resolution. This section is definitely optional, as if you set up the objects properly DesktopX themes should work with different resolutions. There will be cases however when themes need to work at specific resolutions. For example, you may have many objects which won't all fit onto a smaller screen. Where this is the case, then you should check the boxes of all resolutions at which the theme is designed to work. Whilst this will not stop users of other resolutions using your theme, it will at least prompt them that the theme wasn't designed for that resolution, and as such may not look as it was intended to.

Wallpaper and Labels

Via the two other buttons on this tab, you can define additional visual configuration options.

One of the quickest ways of changing the visual user experience is to use a wallpaper just as you would on a standard desktop. If you click the 'Wallpaper' button you can then 'Browse' to find the wallpaper you would like to use. You can use several graphic formats for wallpaper but using a JPG will keep the theme as small as possible. You can also specify the visual style of the wallpaper here, just as you can do in Windows.

You can also choose to use a 'Solid Color' on the desktop by checking this option and selecting a color by clicking the 'Change' button.

If you are using the 'center' wallpaper style you may want to use the 'Solid color' option as well, so you can define the color of the desktop around the wallpaper when the desktop is larger than the image size.

All your objects can have a text label, so DesktopX allows you to set a default for these at a theme level so you can ensure that the color and font is consistent with your theme.

If you click the 'Labels' button you can define the color and font in which the label appears.

In both cases you can choose to use the system defaults as defined in the Windows Display Properties. Alternatively you can custom define them. In this case you can choose the default color of labels and the color when the item is selected.

When selecting a custom font, if you are going to distribute the theme to other users make sure that they will have the font on their machine as well. Alternatively, subject to copyright issues,

you can add the font as a custom file and it will be distributed with your theme and automatically be installed.

Theme Information

When you come to save your theme you can enter some information about the theme to inform the user what it contains and give credit to anyone who has supported you.

2.2 Creating an Object

OK, so we've done the background work and now it's down to the good stuff. The majority of the rest of the Developers Guide is dedicated to objects and all that they can do. What you'll be pleased to know is that the actual creation of a basic object is very simple.

There are several ways that you can create a new object.

On the DesktopX Settings dialog, you can simply click 'New' on the 'Objects' row or the Theme tab.

There are two other ways to do the same. If you right click the DesktopX icon on the system tray you can then select 'New object' from the menu that appears.

Where you have hidden desktop icons in a theme, you can also get this menu by right clicking on the desktop.

When you select this a basic object will appear on the desktop ready for you to customize.

When you do this the 'Object Properties' dialogue will also be displayed to allow you to start configuring your object.

This is where all the fun begins!

2.3 Object Types

To change the object type, click the 'Change' button next to where it says 'Type of object' in the 'Object properties' dialogue.

This will present you with a list of the other basic object types you can use without having to even consider plugins or script.

Once a type is defined, the activity associated with that object type will occur when an object is clicked, though the object's 'Start with' parameter can be changed to alter when this occurs. Doing this is sensible as you can't be sure what the default activation action is defined by the user. More on this later.

We will go through each of these items in turn and explain how to use them. I encourage you to try them out as you go and experiment with the functionality on offer. This is the best way to learn.

Different object types will have a range of options. For a layer, you can choose to make it 'Accept Drag & Drop'.

Layer

The default type of object created is a 'Layer'. A layer object doesn't interact with the user in any way. There are several reasons why you may want objects of this type. It may be a text label providing information to the user, or it may just be a graphic enhancing the visual experience for the user.

Normally, you can drag files from applications (e.g. Explorer) onto the desktop to create a DesktopX object which links to that file. If you select this option for a layer, it means that if you drag a file on top of it, it becomes 'Contained' within that object and moves when the 'Parent' object moves.

Shortcut

A 'Shortcut' allows the object to link to a file or a folder on your machine. If you choose a file (e.g. Notepad.exe) you can add arguments to that link. For example, if the file shortcut was Notepad.exe, you could add the argument 'c:\mydoc.txt' which would launch Notepad and then open the file 'mydoc.txt'. You can also how that application will appear when launched (e.g. maximized).

In addition to the specific paths that you can enter, DesktopX provides several variables that can be used to point to specific directories.

`%exedir%` - returns the executable directory.

`%objectdir%` - returns the object data directory (where custom files are stored).

`%objectsdir%` - returns the full \Objects directory.

`%themesdir%` - returns the full \Themes directory.

So, for example, if you wished to link to one of the default clock objects that comes with DesktopX, you could link to this “`%objectsdir%\Nicer Analog clock.dxpak`”

If you choose a 'Folder shortcut', you can either choose a system folder such as the Control Panel, or you can choose to browse for any folder on your computer. In either case, this will cause an Explorer window to open displaying the contents of that file.

You can however, check the box below the dropdown menu to popup a menu of the contents of that folder. You can then click an item on that menu to run that item directly. You can even specify where the menu appears relative to either the cursor or object itself.

What you should know is that these objects will function like the real target of the shortcut. For example, if you drag the file c:\mydoc.txt to an object which is a shortcut to Notepad.exe then that file will open up in Notepad. Dragging a file to a folder shortcut will move or copy the file to that actual folder.

URL

A 'URL' object type is similar to the shortcut but it is dedicated to launching web pages in your browser, so all you need to do is enter the web address (e.g. www.wincustomize.com).

Object Controller

Objects can interact with other objects and this is a way to carry out some of the most common of these interactions.

An individual object or groups of objects can be defined as 'popups', which means that they can be shown on screen only when they are required. We will define how to do this later, but at the moment it is enough to know that you can select one of the first 2 options here to show or hide the object.

You can also choose to 'Destroy' an object. This might seem somewhat destructive, but is sometimes useful. For example, when a theme is loaded you may want to display a welcome message to the user. You don't want this after it has been read though, so you may set it so it destroys itself when clicked.

Obviously these commands mean nothing without specifying which object is the target of this action. You do this by selecting the object from the 'Target object' dropdown list.

Taskbar

The 'Taskbar' object offers some immediate functionality. This transforms the object into a display of the applications running on your machine.

Obviously there are a few configuration options to adjust how you want the taskbar to look.

You can choose whether the taskbar runs horizontally or vertically, and at which end of the taskbar new items are added, and the size of the icons used to represent the items on the taskbar.

If you want, you can add labels which display the name of the application, and choose the color of the text used dependant of whether the item is selected or not.

The final 4 items really define the structure of the taskbar. The first item lets you define how much space the maximum size of each item on the taskbar. Essentially this defines how wide (in pixels) each item will be where 'Use labels' is selected.

You also need to specify how much space there is between items. The final option allows you to vertically offset the contents of the taskbar. This may be important depending on the graphic used in your object. The graphic is defined like any other except you get a customized 'Advanced' option which you need to make good use of to customize the object. We will get onto this in the appropriate section.

Normally, the taskbar just grows and shrinks depending on the number of items in it, but the final option is a check box that allows you to lock the size of the taskbar. When this option is set the size is defined by the height or width option on the Object Properties 'Summary' tab (depending on the taskbar orientation). This value can be an absolute number of pixels (e.g. 400), or a percentage relative to the screen width (e.g. 80%).

System Tray

Once you have understood the taskbar type, the 'System Tray' class is easy, as it uses a subset of these options that creates an object to display the icons in your system tray.

System

The 'System' class allows you to access some of the other standard Windows functionality.

Selecting this class provides you with a dropdown list of items from which you can select. Each of these items has a corresponding Windows dialogue box that will be displayed when the user interacts with the object.

DesktopX

The 'DesktopX' class provides you will a list of things to do that relate to DesktopX and it's user interface. Each of these items is fairly self explanatory.

Widget

This class is useful if you are planning to export the widget as a stand alone Executable file (or widget). You can use these to replicate the commands that are available in Widgets – to close the widget, minimize it, or display the About box. Widgets are explained in more detail in section 2.5.

2.4 Other Object Basics

The object is now created in a rough form, but now is an opportunity to run through the other basic elements relating to objects before we move onto more complex things.

Labels

The first thing you can do is to add a text label to your object. You simply do this by entering the label text into the 'Label' box at the top of the General tab.

If you want to explicitly set the 'Label color' just hop over to the Relation tab, and at the bottom of it you will see 2 buttons where you can set the color of the label depending on whether the object is selected or not. Clicking the button offers you the choice of using the default color as set in the DesktopX Settings ... Theme Settings ... Labels area, or defining a different color specifically for this object.

Summary tab

If you look at the 'Summary' tab of the object, you will see a range of other parameters that you can change.

Here, you can tweak the objects position and size as well as assigning a 'Tooltip' message that appears when you hover over the object.

In the 'Description' section you can add some information to the object in order to identify yourself as it's creator, and also add any comments that you feel may be useful to new users of the object.

Most of the content of the 'Origin' section we will cover in the Interaction sections, but for now, the 'Object ID' is important for you to learn about.

This is a unique identifier for the object. It is crucial to assign a unique name to the object for a range of reasons. Any time you deal with object interaction, all types of which we will come onto later you need to identify the object you want to interact with so it must be identified using the 'Object ID'.

Even before you get onto object interaction you will quickly understand the need for this when you try to use the 'Object Navigator'. This is a tool that allows you to see all the objects within your theme, along with their properties.

To open the Object Navigator, right-click the DesktopX icon in the Taskbar and then click "Object Navigator".

The Object Navigator will show you all your objects, relationships they have with other objects, position and size, z-order and images used. It is not designed to tell you everything, but rather a summary reference that you can use to identify and tweak objects. There are certain elements of an object's properties that you can edit directly via the Object Navigator for speed; the object position, width, height and comments.

You can select objects using this interface by clicking on them. Use the SHIFT and CTRL keys to select multiple objects. You can also right click an object to access its properties, just as you would by right clicking the actual object on screen.

The row selections on the Object Navigator are synchronized with the actual object selections on screen. This means that at any time you can clearly see which object you are working on.

Imagine the chaos if you didn't use the Object ID property and all the objects were called <unassigned>!

Another useful thing to remember is that when an object is selected you can move it by using the cursor keys to fine tune its position. In order to do this, hold down the CTRL key and click the object(s) you wish to move to select. Keep the CTRL key held down and use the cursor keys to move the object. If you hold down the SHIFT key at the same time then the object will move by 10 pixels at a time in the direction specified.

You should also be aware that as you move an object near a screen border, the object will attempt to "stick" to that border. This is the most commonly required behavior, but if you want to precisely position the object close to, but not on, the edge the above keyboard shortcuts will be the best way to do this.

Alignment

In the summary tab, by default you align an object to the left and top of coordinates specified at the top of the 'Appearance' section. You can however choose to centrally or right/bottom align to these coordinates by clicking the 'Align' buttons and specifying the alignment that you want to use. This applies to System Tray and Taskbar type objects as well as regular image and text objects.

2.5 Exporting Widgets, Objects and Themes

OK, you've learnt the basics of creating objects, but now you need to know how to export Objects and Themes.

First thing to note quickly, is that though it's a great idea to save your work frequently, every time you click OK or Apply on a dialog, the current work on the desktop will be saved and then reloaded next time DesktopX is reloaded.

Saving a theme is quick and easy. In the DesktopX Settings dialog on the Theme tab, all you need to do is click the 'Save As' button. You will then be prompted to specify and name for your theme and choose where you want to save it. You can also specify some additional information about your theme.

At various locations, you may see a 'Save' option along side the 'Save as...' option. This simply saves details of your existing theme to the hard drive so if anything went wrong with your machine, DesktopX would be able to restore to this position when reloaded.

Exporting objects and widgets is also really easy. To do this, the first thing to do is select the object(s) you want to export.

You can do this on the screen by dragging a rectangle with your cursor. A selection area will be highlighted and those objects selected will take on a 'blue tint'.

You can also do this by selecting the objects in the Object Navigator.

Once you've selected the objects you can right click one of the selected objects and Click 'Export...'.

Following this, you will have a range of options. On the left of the dialog, there is a choice of exporting specifically the objects that you have selected or, also exporting and objects related to these (by group or parent/child relationship).

You also have a choice in how you export the file. The first is the most common option. Use this to create a “.dpack” file which is a standard object pack that can be loaded by DesktopX users.

The second export type is as a “Widget”, which I introduced you too in the User's Guide. A widget is a standard executable file that you can run on your computer alongside DesktopX, or even without running DesktopX. You can share this widget with others, but you should be aware that the widget will only run on computers which have DesktopX installed.

After specifying one of these two options, all you do is specify a name and location for the exported file and then it is saved.

Remember that if you create a normal “.dpack” you can upload your work to the WinCustomize website. By doing this others can benefit from your creations and you may get feedback from users that helps you develop in the future.

DesktopX Pro users have a third export option. This is also for the creation of a widget, but this widget will run as a standard program on any machine without requiring that DesktopX be installed.

Selecting this gives you a range of options to complete which are related to the object or are displayed in the ‘About’ box:

Application name: Specify the name for your application here

Company: Place your name here, either yourself as the author or your company

Version: You can specify the application version here

URL: You can provide a website address that the user can link to from the ‘About’ box

Application icon: Optionally you can check the box and specify a “.ICO” file that will be used for the application. If an icon is not specified then the default DesktopX icon will be used.

Run Type: You can specify how you want the application to run. Main applications will probably want to run as “Taskbar items” and utilities run from the “System Tray”.

Multiple instances: Specify this is you want to allow more than one copy of the application to run at any time. For example, if creating an international clock object you may want to allow users to run several instances to allow them to have clocks for different time zones.

On the next page you have options to export the file in two ways. The first way is similar to the standard widget, in that a single executable file is created which can be run. All the files needed by DesktopX are placed in a temporary directory when this is run, so it is still possible to access these via script.

The second export type separates all the files used so they can be installed using another method. You will want to use this option if the widget is part of a bigger deliverable that you want to package using a custom installer.

The final item on this page allows to specify a text file that will be accessible from the “About” box. This can be used to communicate information about the application or a licensing agreement for the applications use.

Once you have specified your choices here, the next and final page asks you to confirm your agreement to the distribution of the executable files, and then you can specify where you want to store your application/files.

Users can choose to have widgets run on startup from the widget’s right click menu.

3

Enhancing Objects

3.1 *Object Appearance & States*

OK, well done, you've conquered the basics. Now let's start making the objects look cool!

There's no getting away from it - this section is a big one in which you will learn a lot!

Don't worry though, you can take it bit by bit, and after this section you will be well armed to create some visually stunning objects.

States

First of all you need to understand what 'states' are. DesktopX does not restrict you to having one way that an object can appear. Under different circumstances the object can have a different appearance. Each different appearance is referred to as a 'state'.

The default state is 'Mouse away' which is how the object looks when the cursor is not over the object. Unless you create any other states, this is how an object will look at all times.

First we will go over the basics of what you can define in a 'state' and then we will return to looking at the creation of additional states.

Image objects

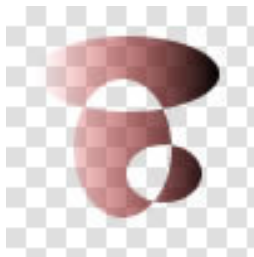
Below the list of states you will notice a series of tabs. The tab shown by default is 'Appearance' so that is where we will start.

An object can fundamentally have two different types of 'Appearance', either 'Image' or 'Text'.

If you want a graphical object, you can choose an image already in the theme by selecting it from the drop-down list, or you can 'Browse' to find a new image on your computer. Images of various types are supported (BMP, ICO, PNG, TGA and JPG).

When you save images they are saved as rectangular images, but obviously you don't want to be restricted to having rectangular objects. Both PNG and TGA images allow you to include

transparency information with the file, which means that although the saved image may be a rectangle, the actual image that you see on your screen can be any shape you want, because the background is visible when transparent areas of the object exist.



This image on the left is a PNG image created in Macromedia Fireworks with a checkerboard effect showing the transparency.

When this image is used in DesktopX all transparency is preserved and the background shows through as you can see on the right.



DesktopX even allows this sort of effect in image formats that don't support transparency such as BMP. In these image formats, DesktopX will interpret the color referred to as magic pink to be transparent. 'Magic pink' is pure pink with an RGB value of 255,0,255.



Although using this method you can't have a graduated transparency as you find in the PNG example, you can make a close approximation. Note, that you need hard crisp edges as if you 'anti-alias' to soften the object edges, your object will have a pink glow as these colors are not pure 'magic pink'.

One final advanced tip for you. If you're forced to use BMP images 'magic pink' for transparency there is a final enhancement you can make to achieve graduated transparency like PNG files can provide.

What you need to do is double the height of your source image and in the lower half use grayscales to indicate the level of transparency. This is best explained by looking at the image on the right.

Black represents total transparency, and the closer you get to white the more the object will appear.

If you use this graphic normally you will see the grayscale part, so you need to go to the transparency tab and check the box that says "The object's transparency is set by the graphic used".

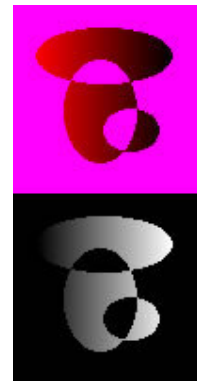


Image sizing and offsets

Once you have selected the image to use you can distort the image by entering a width and height for the image in the appropriate boxes.

DesktopX is not content with sizing objects in such a brutal fashion, so provides an advanced tab to allow you to resize the image more intelligently.

Clicking the 'Advanced' button brings up a new dialog. This allows you to specify different scaling horizontally and vertically. At each edge you can specify a number of pixels that you do not want to stretch and then you can choose to either Tile (repeat) or Stretch the area in between.

For example, if you have an image that is 40 pixels by 30 pixels and you want the outer 5 pixels at each edge to remain unstretched, then the corresponding values A-D would be 5, 35, 5, 25. You could then choose to either tile or stretch the area in the middle.

Also in this dialog, you can offset the image position. This is not useful in a single state object because you could just position the object where you want, but in a multi-state object, you can offset the image in different states to create useful effects. A particular example of this could be offsetting on "Mouse down" to represent the click like a button being depressed.

Text Objects

If you choose a 'Text' object type then the configuration options change to a more simple set.

You can enter the text you want in the text area, and set it's alignment in the dropdown 'Alignment' list. To further customize your text, you can specify the font (including style and size) and color in which the text will appear.

If you want, you can specify that the text has a border by clicking the box and selecting a border color. By having a colored border and setting the text color to 'Magic pink' you can create text which is purely an outline.

You can also make use of offsets here to create effects like you can for images. Click the "Advanced" button to specify the offsets you want to use.

Object Transparency

We will come onto the 'Animation', and 'Messages' tabs later, but the remaining four allow extra tweaking of your image.

The 'Transparency' tab allows you to make the whole object partially transparent (in addition to any transparency specified in the image).

You have a 'No transparency' or you can specify a uniform level of opacity of the image. For example an object with a uniform opacity of 40% will be 40% visible and 60% transparent.

The final option allows you to vary opacity as the object animates but we will discuss that in the appropriate section.

Shadows and Glows

The 'Shadow/Glow' tab allows you to apply either a drop shadow on the object giving it the appearance of floating above the desktop, or a glow surrounding the object.

If you click "Enabled" button you can specify four parameters to customize the shadow.

Setting the offset defines in which direction the effect goes. Setting the two offsets to '0' will create a glow, whereas non zero values will create a drop shadow in a direction dependant on the positive or negative values.

A low 'Sharpness' value gives a crisp edges shadow whereas a high value makes it softer. The higher the 'Darkness' value the darker the shadow will be.

You can also specify what color you would like the shadow or glow to be by clicking on the appropriate button.

This feature can be effectively combined with shadows to create great special effects.

Color

One of the coolest features of DesktopX is the ability to change the color of things without replacing the image. In the 'Color' tab you can adjust the objects hue (color) by selecting a value between 1 and 255. Experiment until you find a shade you like! Setting the value to 0 will set the object color to that of its original image.

You can further fine tune the color by tweaking its brightness and contrast across the specified ranges.

We will come onto additional states later but note that there is a button here to quickly apply the change you have made to all different states of the image.

Sound

The final tab that we will cover in this section is the 'Sound' tab. Here you can specify that a sound is played when the state occurs simply by browsing to it on your hard drive. You can Browse for or select an existing WAV or MP3 file.

Although we will discuss animation later, you can see options for looping the sound when the state is animated. You can choose to repeat the sound once with every loop of the animation, loop continuously irrespective of where in the animation you are, or not loop at all.

New States

You now know the basics of configuring a single state object. A multi-state object is no more complicated.

To add a new state, simply click the 'Add' button. You will then be presented by a list of the common states not yet used from which you can select:

Mouse over - occurs when the user moves the cursor over the active part of the object (as defined by the 'Activation' option on the 'Relation' tab.

Mouse down - occurs which the user presses down the left mouse button over the object

Mouse up - occurs when the left mouse button is released whilst over the object

Show - occurs when the object appears on screen (either when loaded or via script)

Hide - occurs when the object is removed from the screen (either when unloaded or via script)

Command executed - occurs when the object is activated as defined by the 'Starts with' option on the 'Relation' tab.

You can then configure this state just as you did for the first one. One thing to remember is that all the images used must be exactly the same size.

The other buttons in this are should be fairly clear as to what they do. Obviously the 'Remove' button removes a created state though you obviously need to have one state for an object.

Once you have spend time setting up the first state, the likelihood is that other states will be fine tuned versions of that state, so for speed you can use the 'Copy From' button to copy the settings from another created state to the currently selected one.

The 'Reset' button clears all modifications in the currently selected state so you can start again.

When you create a state the drop-down list shows you the states that exist by default in DesktopX, but you aren't just limited to these. In the drop-down list box you can just type any name in and a 'Custom State' of that name will be created which can be configured just like any other.

Though these states are not triggered by user interaction they can be activated by Messages and Script which we will come onto later and are very useful for that purpose.

The one final thing to define in a multiple state object is the 'Default Appearance'. On the 'General' tab you can specify the image to be used when the object first loads. This will not change until the object is interacted with in some way. Note that this image must be the same size as all the other state images.

3.2 Animating Objects

Whilst everything you learnt in the previous section was great, there's nothing that catches the eye like movement, so DesktopX makes it easy for you to add animation to your objects.

The 'animation' tab is where you can define how your object animates.

The first thing to explain is that there are the two different types of animation that DesktopX can do.

A 'Static animation' is the most simple animation option. This simply changes the level of transparency rather than the image itself.

Slightly more complex is a standard animation with different images over time.

Static Animations

We will start with a 'Static animation'. To start with, set a number of frames. A frame is an individual part of an animation, so the higher the number of states the more different 'images' an animation has. Once you set this value higher than '1', the 'Interval' is enabled. This is how long each part of the animation lasts in milliseconds. The higher the value the longer the animation takes. For example, a 5 frame image with an interval of 50ms will take 250ms (1/4 of a second). You also need to check the 'Static animation' checkbox so that DesktopX knows not to expect an animated image, just to vary transparency.

If you check the 'Loop' box then the animation will be repeated as long as the object is in that state. Normally if the state changes then the animation will play through to the end, but if you check the 'Interruptible' box then to state will change immediately without playing to the end.

After that you simply need to set the direction in which the animation will fun by selecting one of the 4 options available.

If you are looping the animation it will probably make sense to choose one of the bottom pair as this will lead to a nice smooth pulsing transparency effect.

The final thing that you need to do is specify the change in transparency. This is where you get to use the third option on the 'Transparency' tab. Here you specify how the transparency varies in one cycle of the animation. This will then be impacted by the animation style you chose and whether you chose to 'Loop' the animation.

Try experimenting with these options and see how they change the appearance of the object.

Image animations

We will now look at actually animating the image itself, rather than just animating the transparency.

Though I said that this is more complicated, it still isn't actually that difficult. The change you need to make is to combine all the frames of the animation into your source image.

What you need to do is place the frames of the animation side by side from left to right.



The above is the source image for a six frame animation of the DesktopX logo being worked on in Fireworks. There are six 100x100 images next to each other in one 600x100 image.

If you run this forwards then the image will increase in size.

Once you have done this then just change the other settings as you were shown for the 'Static animation'. This includes transparency so you can change both the image and the transparency over the course of an animation.

3.3 The Basics of Relationships

You've already learnt about how the objects you create can interact with the user, and will now learn how they interact with each other.

DesktopX allows you to tweak how they interact with users, but also how they interact with the desktop and other applications on screen.

To access these settings, open up the Object Properties dialogue and then click the 'Relation' tab.

Visibility

The first thing you can specify is whether the object is visible or not. You may wonder why you would want to hide objects. There are two main reasons. The first is that the object may be doing work in the background such as loading a webpage which you don't want visible. Alternatively, you may only want the object to be visible at certain times which are controlled by script or popups which we will come onto later.

Z-Order

If you consider a desktop, then all objects on there have an 'x' and 'y' position which defines where they are on the screen. What you might not consciously have thought about is their 'Z-Order'. This defines which object or application appears 'on top' of the other when they overlap. Objects with a higher Z-Order appear on top of those with a lower Z-Order.

You have three Z-Order options for your objects. The first is 'Desktop'. This means that running applications will always appear above your objects. They will open on the desktop. Those interacted with most recently (clicked/moved), will be on top of other objects.

A 'Normal' Z-Order means like the object will behave like a mini-application. Objects with a 'Normal' Z-Order appear above or below applications and other 'Normal' Z-Order objects depending on which were interacted with most recently.

An 'Always on Top' Z-Order means that this object will always be visible above all applications and objects. It functions in the same way as the 'Always on top' setting of the default Windows taskbar.

Movement

'Movement' can be defined as 'Normal' or 'Locked'. If it is 'Normal' then the user can drag the object around the screen. If 'Locked', this is not possible. The user can still move the object by holding down the CTRL key and dragging, but accidental movement is not possible. You can also choose the 'Default' mode, which means that its movement is defined by the User's DesktopX Settings as we discussed in the last section.

Activation

'Activation' defines how the user can interact with the object. If the setting is 'Default' then it simply follows the user's settings, but you have three other options. 'Rectangular' activation is based on the size of the source graphic and ignores any transparency in the object, whereas 'Visible Area' means that only those areas of the object visible on screen can be interacted with. The final option 'None' means that the user cannot interact with this object and any attempts to do so are ignored. This is usually most important for cosmetic objects on the desktop.

Start with

'Start with' defines the way in which a user interacts with an object. It is not always the case that you want a user to have to click an object to activate it. In addition to the standard 'Default' option, you have four more ways in which a user can interact with the object. Sometimes for example you may simply want the user to move over an object for something to occur.

Default Cursor

This option allows you to specify which cursor will be displayed when the mouse moves over the object. This is a very useful way to indicate to the user the effect of their interaction with the object, for example if you want a user to drag and move the object rather than clicking it then it would make sense to use the 'Move' cursor.

Popups

OK, now onto 'Popups'. These are a really cool feature in DesktopX that we briefly mentioned in the Object Types section.

Earlier in this section we discussed visibility. You don't always want objects to be visible on the screen, and in some cases you may want them to be 'Popups' whose appearance on screen is controlled by other objects that have the 'Object Controller' type.

If the object is referenced by an object controller then it will function as a popup depending on the value set here. If the object is not used as a popup in this way, then this value is simply ignored.

You should note that when an object is called as a popup then all children of that object or objects in the same group will also inherit the popup functionality of that object. This is because popups are just really a shortcut for the visibility setting. By this, I mean that popups are essentially a dynamic way of setting the visibility of the object. Visibility is a property that is shared by group members and children inherit it from their parent. Don't worry if you don't understand the parent/children definition. We discuss them in more detail later.

Here are the popup types.

'Static' popups - Normally, an Object Controller will toggle the appearance of a popup (open it or close it depending on whether it already is visible). Once these 'Static' popups are opened they

must have a different Object Controller to the original one to close them. Think of them like a normal program. Clicking a link to a program will open it, but clicking the link again will not close the application. A separate 'Object Controller', the close button, is required to do this.

'Toggle' popups - Are the simplest types. If the popup is hidden when you click the Object Controller it will be shown. If the popup is visible when you click the Object Controller it will be hidden.

'Menu' popups - These popups once displayed, will only remain until the user interacts with the desktop in anyway. They have this name because they function just like menus; unless you click them immediately to carry out the function they are there to do they disappear.

'Volatile' popups - Once a 'Volatile' popup is shown, it will remain until either the controller is toggled or another popup is opened.

'Volatile (No toggle)' popups - This is just like the previous version, except that toggling the Object Controller is disabled like in a 'Static' popup, so the only way to make it disappear is to activate another popup.

Position Adjustment

When you are developing for yourself you will design everything so it fits your screen perfectly. So, what happens if you change your screen resolution or you give your work to another user who runs at a different resolution?

Well, DesktopX can make a fairly good guess but it is certainly not perfect, so if you are going to make good objects you should pay attention to the two options you see above.

By specifying values in here, you can define how the object will reposition itself when the screen resolution changes. DesktopX logs what resolution the screen was designed on and therefore detects changes and reacts according.

The 'Dock to Left' and 'Dock to Right' options mean that the object will remain the same distance from that edge as it was when originally designed.

'Dock to Nearest xxxx Border' means that the object will align itself to the nearest edge of the screen (vertical or horizontal depending on option). For example, if you have an object against the tight edge of the screen then increase the resolution, you run the risk that it will end up floating away from the new edge.

An object that repositions based on the 'Center' option will be relatively the same distance from the center of the screen as it was in the resolution for which it was developed.

The final option 'Rescale' means that the object will be relatively the same distance across the screen at different resolutions. For example, an object positioned at 200,200 on an 800x600 screen would be positioned at 320x341 on a 1280x1024 screen (25% across, 33% down).

3.4 Groups, Parents and Children

When developing you will soon realize that many of the best objects are created by several objects working together. For example, consider a simple 'button' made up of a single image. Whilst this may be a good object, it's not very flexible. For example, if you want another similar button you have to create an entire new graphic; text on the button cannot be changed dynamically, the font cannot change, if you share the object it may not be relevant.

Now, consider creating that button from a background image and a separate text object. All of a sudden it is much more flexible as the text element can easily be changed and all the above problems are gone.

One problem that remains however is the fact that you need to ensure that these objects work together as a single unit. Groups, Parents and Children make this very easy.

If you want several objects to work together is to assign them to the same 'Group'.

Groups

An object's Group is defined on its "Summary" tab. Any predefined groups are available in a combo box so you can allocate an object to that, or you can simply type in a group name to create a new group. Alternatively you can select one of more objects, right-click and click Group from where you can get the same options.

Once a group exists, objects within that group share some basic functionality.

All objects in a group move together, so as you drag one, all other objects in the group move at the same time. Note that you can hold down CTRL when dragging to just move that single object.

Also, as we saw earlier, popups work well with groups. All objects in a group popup or hide together.

Note that you can remove an object or objects from a group by clearing the content of the 'Group' field in its properties.

Parents and Children

'Parent / Child' relationships are slightly more intimate in that an object is more directly affected by changes made to its Parent/Owner. Again, in the same section there is a 'Parent/Owner' drop-down list where all the other objects in the theme are listed. Simply select one of them and your current object becomes a child of that object.

If a Parent is hidden or moved then the same is applied to objects that it “owns”. Owned objects are also shown higher in the z-order than (above) its Parent. You might initially think that this is excessive when Groups work in almost the similar way. Consider however, that this saves so creating many named groups to manage where they are unnecessary, it provides increased control over the z-order and also that it provides increased flexibility to control visibility within a group without affecting the whole group.

One added option you have for an object that is owned is to make it a 'Child'. This means two things. The first is that its position is defined relative to the Parent object not the screen. This makes positioning within the parent easier. Also, the object is only visible while it is in the confines of the Parent object. Note that this is based on the actual rectangular size of the object, not defined by the visible area.

Children also make DesktopX more efficient and therefore should be used wherever possible.

3.5 Sending Messages

Sending messages in DesktopX is actually a really simple concept to understand and simple to achieve.

In summary, sending messages means that when a user interacts with one object, cause one or more different objects to react. Consider a light in your room. When you flick the switch it changes state, but also the light bulb itself changes state. When you flick the switch, a message is sent to the light bulb which also changes state.

Sending messages is very simple. Simply, go to the 'States' tab of the object you want to send then message from, select the state which you want to occur in order for the message to be sent (e.g. Command executed) and then click the 'Messages' tab.

Once you have done this click the 'Add' button to pop up the messages dialogue.

There are two things to define in this dialogue; the Destination and the Message to send.

The 'Destination' is the object to which you want to send the message. For example in the example above the object to which we would be sending the message is the Light Bulb.

You have two options here. The first is to send the message to every other object in the theme. You should use this option very carefully as obviously in a large theme this could cause several objects to change state at once and you may end up with a massive cascade of state changes, or even a loop you can't get out of if objects send messages to each other.

The more likely option is that you will be sending a message to a specific object, in which case you should select that option and choose the object from the list.

The second thing to define is the message itself. The drop down list will show a list of the default messages from which you can select. Alternatively, if you know of a custom message that has been set up for the destination object you can send that message by typing in the name of that message.

You can add as many messages as you want and an object can send different messages in different states - e.g. it may send one message when the mouse is over an object, and a different one when the mouse moves away from the object.

That's it - sending messages is that simple and the results can be really impressive.

3.6 Custom Files

In the summary tab, there is a button labeled “Custom Files”. This allows you to attach files to objects just as you would do to email that you send.

The first reason that you may wish to do this is to attach fonts to a package. If you have text objects that use a font that you are not sure a user has on their machine then you can attach them as a Custom File.

DesktopX automatically installs any fonts that included with a pack, so then you text objects will appear as intended.

This introduces an important point. You must ensure that by attaching Custom Files of any sort that you have the file author's permission or that the file is not subject to any copyright or licensing agreement which does not permit distribution.

There are many other potential uses for Custom Files. When a package containing Custom files is loaded they are automatically extracted to the theme directory so they are accessible from there.

For example, if you attach your brochure as a PDF you could create an object that links to it using the shortcut “%objectdir%\Brochure.pdf”

You may want to attach a flash movie which can then be loaded into a web browser control.

Budding musicians may want to attach MP3 files and create a “demo track” object that they want to send to record companies.

The options are unlimited.

3.7 DesktopX Plugins

DesktopX is a flexible program that opens itself up to developers who can create plugins. These are additional modules providing packaged functionality that you as a user can access.

To create an object based on a plugin simply create an object as normal, then in the 'Additional Abilities' section of the 'General' tab, click the 'Add' button. This will then pop up a list of available plugins on your system.

All you need to do is select the plugin you want to use, and then click OK.

Each plugin needs to be configured differently but we will go through a few examples of the different types here.

If you need to remove a plugin, simply select the plugin in 'Additional Abilities' and click 'Remove'.

Digital Clock



This plugin when used immediately creates a digital clock on your desktop.

In a plugin the developer can create customization features to tweak the plugin. If you select the plugin on the 'Additional Abilities' list the 'Configure' button will be enabled.

Clicking this allows brings up a dialogue which allows you to change the time format and the font and color in which it is displayed.

Digital Clock (Text State)



This plugin shows a different way the developer can implement the same type of functionality but give more flexibility to the user. When this is used the immediate impression may be that nothing has happened.

However, if you go to the 'States' tab and check 'Text' on 'Appearance' then all of a sudden you have a digital clock on your desktop. What's cool about this implementation is that it uses standard DesktopX text so you can tweak the text any way you like including shadows, and it is exposed to manipulation via script such as the ability to change its color dynamically.

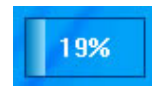
Email Notify



You can configure this plugin by entering your mailbox details and how often you want to check. The way it works it to create two new Custom states (you recall we mentioned these earlier?) called 'New mail' and 'No new mail'. You can define each of these states to be representative of whether you have mail or not. These could be images or simply different text.

You may even decide that it makes sense that rather than this object being a 'Layer' it is a Shortcut to your email program.

Performance Meters



This delivers dynamic graphics to your desktop that would be difficult to create otherwise.

You can chart CPU and other performance measures in a range of graphs and a range of colors, all accessible via the 'Configure' button. Tweak to your hearts content! Remember that 'Magic pink' provides transparency. Also remember that 'magic pink' doesn't work in gradients because most of the pink in the gradient isn't 'pure' and hence 'magic'.

DX Player

This delivers complete interaction with your media player be it ObjectMedia, Winamp 2.x or CoolPlayer.

You can define what happens when the user interacts with the object, even to the degree of whether they are holding down any keys.

In some cases images are needed and you can define them here as well such as gauges for volume etc.

I'm not going into the detail of the plugin here. Check out the 'Sputnik' player that comes by default with DesktopX to see what can be achieved with this plugin.

4

Tips and guidelines

Before using an object you created, and most important before releasing it to other users, be 100% sure that everything is perfect and polished. Below there are a few tips to help you do this.

4.1 Do not assume Default “Start with” command

DesktopX global preferences let you choose between single click and double click default action for objects with an associated command. This is a local user setting.

The Relation panel in the object properties panel lets you override the default activation mode. So, if you leave Default, the global user setting will be used. If you set it Single or Double click, you'll force activation in that specific mode.

Here is where you should be careful about consistency. Except for DesktopX ‘Shortcut’ objects, it is recommended that you specify the required activation behavior.

For instance, buttons normally only need to be clicked once, so buttons of an object such as DesktopX media player objects need to have “Single click” activation. It could appear to you to be fine as Default, because you can have Single click default global settings, but the object will be basically broken in its functionality when distributed and loaded by users with Double click global preference.

4.2 Plugins

Starting with latest DesktopX 2 builds, the plugins issue has become more important, and a black list is now in action (it is located in the DesktopX directory and can be edited to add or remove listed plugins).

A first distinction is between so called "Old-style" plugins and "SD plugins". The old style plugins were built around the first very old plugin interface of DesktopX. The support is still in DesktopX 2 but only for backward compatibility. These plugins are stored into the \DesktopX\Plugins\ folder, and the objects using them have a slightly different GUI. They are also recognizable because a warning message appear when editing them: "This object is using old-style plugins...".

It is strongly suggested that you don't use these old plugins for creating new objects and if possible, update old objects to use new plugins. They can cause instability and maybe not be supported in the future.

The plugins you should use are the ones built around the SD plugin specification, compatible with DesktopX and ObjectBar and documented in the Developer section. Still, there have been buggy plugins around that have forced us to blacklist them. On loading themes using them, you have the option to continue loading or avoid that they are loaded.

Such plugins like DXToys, while providing some very cool features, could cause instability, memory leaks, hang DesktopX and most important are often incompatible with the newest Version 2 features.

We are in the process of updating the existing "safe" plugins and replacing the functionality removed by blacklisting plugins with new updated plugins.

The only plugins you should use are the following ones. All other are old, buggy or obsolete.

- "Performance Meters 2" (DXPerf4.dll) – Performance monitoring.
- "DXPlayer" (DXPlayer.dll) – Media Player features.
- "Trash bin" (DXTrash.dll) – Updates object states accordingly to recycle bin status.
- "Analog Clock 2" (SDAnalogClock2.dll) – Antialiased clock arms.
- "Analog Clock 3" (SDAnalogClock3.dll) – Bitmapped clock arms.
- "Date (Text State)" (SDDateText.dll) – Sets the text of an object to the correct date. It needs a text object.
- "Digital Clock (Text State)" (SDDigitalClockText.dll) – Sets the text of an object to the current time. It needs a text object.
- "Email notify" (SDMailNotify.dll) – Updates object states accordingly to incoming mail.

- “Charts and graphs for scripts” (DXChart.dll) – Used by scripts to show statistics in a graphical way.

These plugins are distributed with the latest DesktopX2 distributions. If you have doubts about which plugin to use, want to report bugs, or suggest new plugins or updates, feel free to write in news.stardock.com/stardock.desktopx.

4.3 Child Objects

One of the best new DesktopX 2 features is the ability to configure Contained children. You find that option in the Summary panel of Object Properties.

In DesktopX 1, child objects weren't actually ... children. That's because they could be placed outside the parent's area, so the Parent relation was simply a z-order rule. i.e. a child is always above its parent and when you drag the parent, children are dragged together.

DesktopX 2 introduces “real” children, called “Contained” children. The differences are:

- They are contained in the parent area, i.e. if you drag them outside, they will be clipped and not be visible.
- Their coordinates are relative to the parent's origin. This makes things more logical, especially when using scripts, and fixes a lot of problems with screen resolution adaptation, because only the parent will need to be repositioned when loading objects on different screen resolutions, and children will be placed accordingly. The only limitation, at the moment, is that they don't support rules to adapt themselves to changing parent size. i.e. if they are inside a big parent "bar" object, as large as the screen, they won't be repositioned accordingly when the resolution changes. You can workaround this by using non-Contained panels, children of the Bar object, and have Contained children in them.
- They give a huge performance boost! If you have one parent with 20 contained children, that'll count performance-wise as a unique object. This is especially important for Normal and Always on top level objects. Alternatively, one parent and 20 non-contained children will be like 21 separate objects. This significantly impacts on performance at Normal level, and decrease it even at desktop level.

To summarize: use only non-contained children if you really need them outside the parent area (but consider Grouping instead), or need some special behavior for multiple screen resolutions support.

DesktopX2 will currently default to non-contained for backward compatibility, but we are moving in new builds to automatically enable Contained by default when possible. 'Set parent' and 'Group' right-click actions already support this: if you select several objects and Group them, DX will choose automatically the best settings for grouping and children. After that, you may still want to check manually how they are configured.

4.4 AniUtil with Optimization for making animations

AniUtil is a little utility distributed with DesktopX that let you attach single frames into one "animated" image. It supports bitmaps and PNG. One option is Optimization. It will basically analyze and cut image borders that are not used by any frame. This is very important, for instance, when rendering from applications like 3DStudio, where setting as output the minimum image size is either very difficult or impossible. AniUtil will take care of optimizing the final animation making sure the least memory is used.

As advanced option you can even set Tolerance a bit greater than 0, to make sure "dirty" pixels as result of the rendering don't cause the image size to stay big.

4.5 32 bit image formats

For performance reasons, DesktopX internally always uses 32 bit images. That means RGB and a dedicated Alpha channel. So, even if you make 8 or 16 bit images, DesktopX will convert them to 32 bit (and therefore increasing longer load time!). The only apparent benefit of using lower bits images is to save disk space. However, nowadays PNG can provide a great compression level, and the exported DX objects and themes are compressed anyway. So, the suggestion is to always use 32 bit images as BMP, TGA and PNG, for best quality.

4.6 Screen resolution adaptation before distributing your objects

In the Relation configuration panel of Object Properties are options to control the reposition of objects on changing screen resolution and on loading objects on different screen resolutions. It is suggested that you experiment changing your screen size and see how objects behave before distributing them.

Also, when using scripts, you should always use the VScreenX properties, not the ScreenX ones. This is because virtual coordinates support multi-monitor setups.

4.7 JPG wallpapers

When possible, you should use JPG wallpapers in your DesktopX themes. If their compression level is correctly tuned, they can preserve the original quality at a unnoticeable level, but will use a lot less disk space, thus making the themes a lot smaller and let them use less bandwidth.

Note that DesktopX will internally convert them to Bitmaps, so that will seamlessly work on all versions of Windows without the need of enabling Active Desktop.

4.8 Meter objects

Be aware when building objects that show and update information (i.e. using the DXPerf4 plugin or browsing information from the internet or checking the mail with the mail notify plugin).

Too frequent updates of many objects at the same time could cause slow downs, so it is very important to tune every information object carefully. I.e. you don't want a disk space meter object to update every 200 ms!! Also most other meters (CPU, network traffic, etc) should just work fine with an update interval great than 600 ms.

4.9 Writing “proper” scripts

Using the DXScript facility combined with the VBScript simplicity is great and powerful, but be aware to avoid writing “poor” script. It is very easy to build in subtle mistakes that will then cause serious stability and performance problems.

The two most common mistakes are:

You instantiate an object and you do not release it afterwards. This will cause a memory leak.

You use the DesktopX.Object(“foo”).property too often. If you reference the “foo” object frequently, first create a reference to it. This will optimize the script performance, but be sure you DELETE the reference when you don't use it anymore (see above)!!

Always assume that attempts to access COM objects functionality may fail and build in support for handling this. Some users could just lack certain COM objects. Also, objects using scripts to retrieve information from the internet could not have an always-on connection, the website could be down etc. So, you should add runtime checks to avoid script errors and continue with a clean execution even if there are faults.

4.10 Building “proper” objects browsing the internet

Example: Weather Information objects.

These objects are the most difficult to write properly so they deserve particular attention.

You just can't assume that a user is connected to the Internet, so browsing could fail and cause very annoying script errors. You should ALWAYS check the connection status, and also check about the success of browsing operations. You should assume that you can retrieve NULL or invalid data.

Also, objects that connect to the Internet should follow a specific usage protocol to avoid annoyances. They should keep an internal connection status that just informs the script about “Can I try to access the Internet? YES/NO”. This is different from the actual connection status,

in fact it is more similar to the Internet Explorer Online/Offline status, that's independent from the actual network connection status.

A tentative protocol works like this:

- Have a global script variable, e.g. `IsConnected`.
- When `IsConnected` is changed to `TRUE`, the object can update and retrieve info from the Internet. It can also start a timer to periodically do that.
- When `IsConnected` is changed to `FALSE`, the object won't try to retrieve info from the internet. Any timer that does that is "killed".
- `IsConnected` can be first set on `OnScriptEnter` by checking for the actual connection status by calling `System.InternetConnected` and setting `IsConnected` accordingly.
- When `IsConnected` is `FALSE`, the objects GUI should represent this status. I.e. you can use a text object that shows a red "Offline" text. From here on, only the user can manually retry to set `IsConnected` to `TRUE` (i.e. by clicking another DX object). Here is another `System.InternetConnected` check and `IsConnected` is set accordingly.
- The user can manually set the Offline status as well. I.e. one could just not want an object to keep updating from the Internet.

4.11 Never use the Sleep method in scripts

Though there is support for a simple `Sleep` method, it is highly suggested that you avoid it, because it could cause the input to hang.

In most cases a timer can do the job. Though a bit more work and a slightly more complex script is required to do this, it is the way to go if you want to build great objects.

If you are using ActiveX controls, be sure you use the clean ActiveX events to be notified, and not brutal `Sleeps`. For example, in a Web Browser control use the `Sub Control_DocumentComplete` and `Sub Control_NavigateError` events to act according to navigation success.

4.12 Only package useful objects

When exporting objects or theme, be sure you first check Object Navigator the list of objects to be exported. It is possible that you export unused, invisible objects that remain from old work, thus wasting memory and disk space. A careful look at Object Navigator to double-check that all objects are used, that children have proper settings ("Child" = yes), etc. is surely worth the extra effort. Also ensure that all scripts which should be enabled are enabled.